# A two phase hyper-heuristic approach for solving the Eternity II puzzle

Wim Vancroonenburg[1], Tony Wauters[1], and Greet Vanden Berghe[1]

CODeS research group, KaHo Sint-Lieven, Gebroeders Desmetstraat 1, 9000 Gent, Belgium
wim.vancroonenburg@kahosl.be

## 1 Introduction

In this paper we introduce a two phase hyper-heuristic search method for solving the Eternity II puzzle. Eternity II is a challenging money prized edge matching puzzle. Solving the puzzle has been shown to be NP-complete [2]. Hyper-heuristics [1] are a recent trend in heuristic algorithms. They tend to be more general methods than meta-heuristics for solving optimization problems. Hyper-heuristics operate at a higher level of abstraction than meta-heuristics. A hyper-heuristic manages a set of low level, problem specific, heuristics without knowledge of the problem domain, and tries to apply them to the problem in a meaningful way. The two phase hyper-heuristic method introduced here, uses different fitness functions and acceptance criteria for each phase.

## 2 Problem description

We describe the edge matching puzzle as follows. The puzzle consists of a board with $r$ rows and $c$ columns. Thus, the board has $n$ ($= r \times c$) positions. The board has to be filled with $n$ square tiles. Each tile has one colour for each side. The different colours are numbered from 0 to $z$. Tiles can be rotated 90, 180 and 270 degrees. The goal is to put all tiles on the board and give them the correct rotation so that the number of adjacent edges with the same colour is maximized, and that all edges with colour 0 (grey) are on the border (hard constraint).

## 3 Objectives

We studied the effect of using several different optimization objectives. Something similar was done in [4], where multiple objectives were combined into one objective and used as a fitness function for a Genetic Algorithm. We considered the following objective functions:

The first objective ($Obj_1$) is the actual objective we want to maximize, i.e. maximizing the number of matching edges. The second objective ($Obj_2$) is to maximize the number of square regions of 2 by 2 tiles that match in their adjacent inner sides. The third objective ($Obj_3$) is to maximize the number of tiles that have their four sides matched with the adjacent tiles. Finally, the fourth objective ($Obj_4$) is to maximize the number of square regions of 3 by 3 tiles that match in their adjacent inner sides.

We ran some experiments with these objectives using a hyper-heuristic algorithm (explained in Section 4) for a fixed number of iterations, starting from a random solution. An improving or equal acceptance criterion was used. Several problem sizes were tested($10 \times 10$, $12 \times 12$, $14 \times 14$, $16 \times 16$). The results reveal that optimizing on $Obj_4$ results in a higher average number of matching edges. We also noticed that the difference between $Obj_1$ and $Obj_4$ becomes larger for larger problem sizes, in favor of $Obj_4$.

## 4 Two phase hyper-heuristic

In our approach we use a common hyper-heuristic framework. It consists of an iterative framework with two sub-mechanisms: a heuristic selection mechanism which uses some strategy to select a low level heuristic for generating a new (partial or complete) solution in the current iteration, and an acceptance mechanism to decide on the acceptability of the new solution.

During our experiments we noticed that, if we use the best solution (highest number of matching edges) that occurred during a search with $Obj_4$ as input to a search with $Obj_1$, the second search is mostly able to improve the first best solution. This is why we apply our hyper-heuristic method in two phases. The first phase with a duration of $t1$ uses $Obj_4$ as optimization criterion, while the second phase with a duration of $t2$ uses $Obj_1$.

In what follows we discuss the heuristic selection and acceptance criteria used.

### 4.1 Heuristic selection

A simple random heuristic selection method was used to select the low level heuristics. Although more complex selection methods exist, we found that this simple selection method is sufficient given the small size of the set of low level heuristics.

### 4.2 Acceptance criterion

As for the acceptance criterion in our hyper-heuristic method, we found out that the recently introduced **Iteration Limited Threshold Accepting (ILTA)** [3] criterion performs very well compared to other acceptance criteria (e.g. Simulated Annealing, Only Improving, Improving or Equal, Late Acceptance). ILTA is similar to improving or equal acceptance but it can accept worsening moves after a number of consecutive worsening solutions ($k$). These worsening moves are checked if they are still good enough to accept (i.e. within a certain range $R$ of the current best solution's objective value). Its two parameters are more intuitive and thus easier to fine-tune compared to e.g. Simulated Annealing.

### 4.3 Starting solution

To generate an initial solution for the hyper-heuristic search method, we use a simple linear depth first search (DFS). It operates from left to right and from top to bottom, and it is terminated after a certain amount of time. Positions on the board that have not yet been assigned a tile, are assigned one of the remaining tiles at random. For a board of size $16 \times 16$ the DFS obtains on average an initial solution with 370 matching edges (out of 480) in 10 seconds. Our experiments indicated that starting a search from a solution obtained by DFS results in a higher number of matching edges compared to starting from a random solution.

### 4.4 Low level heuristics

In our hyper-heuristic method we use several neighborhoods of the current solution as low level heuristics. Tournament selection (cfr. Generic Algorithms) is used to sample several neighbouring solutions, of which the best neighbouring solution is used as result of the low level heuristic.

**Swap and rotate heuristic**
A first and simple heuristic is to exchange two pieces and possibly rotate them. We constructed 3 versions of this heuristic: one for the corners, one for the border and one for the inner tiles.
**3-exchange and rotate heuristic**
This heuristic exchanges 3 tiles and possibly rotates them.
**Double tile rotate heuristic**
This heuristic rotates 2 tiles at once.
**Partial optimal placement heuristic**
This heuristic is the one used in [5]. They call it a very large neighbourhood. The heuristic removes $n$ non-adjacent tiles away from the board, creating $n$ holes. Then, the tiles are reinserted in to the holes optimally by solving an assignment problem. We created 2 versions of this type of heuristic, one for the corner/border tiles and one for the inner part.

In all the low level heuristics corner/border tiles stay at the corners/border, and inner tiles stay in the inner part. Corner and border tiles are also automatically rotated with colour 0 to the outer side.

## 5 Results

All results were obtained on a desktop pc with (CPU: Intel Core 2 Duo 3Ghz, Windows XP SP3, Java 1.6 JRE). As per the competition rules, we performed 30 runs with a predetermined time ($T$), for each benchmark of a certain size. Table 1 shows the results of these benchmark runs. The mean, median, minimum, maximum and standard deviation of the number of matching edges over 30 runs are reported. We also report the average number of evaluations and the time needed to obtain the best over all solution. The following algorithm/configuration was used: initial solution with DFS, first phase hyper-heuristic with ILTA ($k = 500$,$R = 1.15$) acceptance for time $t1 = \frac{3}{4} \times T$, second phase hyper-heuristic with ILTA ($k = 500$,$R = 1.15$) acceptance for time $t2 = \frac{1}{4} \times T$. All reported neighbourhoods were used as low level heuristics.

| | $10 \times 10$ | $12 \times 12$ | $14 \times 14$ | $16 \times 16$ |
|---|---|---|---|---|
| Predetermined time (s) | 1200 | 1800 | 2400 | 3600 |
| number of total edges | 180 | 264 | 364 | 480 |
| mean | 170.17 | 251.27 | 344.00 | 456.37 |
| median | 170 | 251 | 344 | 457 |
| min | 168 | 249 | 342 | 445 |
| max | 172 | 253 | 346 | 459 |
| standard deviation | 1.02 | 0.94 | 1.11 | 2.58 |
| average number of evaluations ($\times 10^7$) | 7.10 | 6.02 | 3.27 | 17.2 |
| time for best over all (s) | 7.78 | 317.66 | 418.89 | 1004.99 |

**Table 1.** Results for 30 runs with the given time limit

## 6 Conclusion

In this work we describe a two phase hyper-heuristic method for solving edge matching puzzle problems like Eternity II. Each phase optimizes on a different objective. The first objective is a new $3 \times 3$ squares objective which obtains better results than the default objective (number of matching edges). We obtain results of up to 459 correct edges out of 480, for a board of size $16 \times 16$.

## References

1. Edmund Burke, Jim Newall Graham Kendall and, Emma Hart, Peter Ross, and Sonia Schulenburg. *Hyper-Heuristics: An Emerging Direction in Modern Search Technology*, chapter 16, pages 457–474. Springer New York, 2003.
2. Erik D. Demaine and Martin L. Demaine. Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics*, 23(s1):195–208, 2007.
3. Mustafa Misir, Tony Wauters, Katja Verbeeck, and Greet Vanden Berghe. A new learning hyper-heuristic for the traveling tournament problem. In *Proceedings Of The VIII Metaheuristics International Conference*, 2009.
4. Jorge Munoz, German Gutierrez, and Araceli Sanchis. Evolutionary genetic algorithms in a constraint satisfaction problem: Puzzle eternity ii. In *IWANN 2009, part I, LNCS 5517*, pages 720–727, 2009.
5. Pierre Schaus and Yves Deville. Hybridization of cp and vlns for eternity ii. In *JFPC08*, 2008.